WHAT IS CLAIMED IS:

1. An apparatus for data storage comprising:

a cluster of NFS servers, each server having network ports for incoming file system requests and cluster traffic between servers; and

a plurality of storage arrays in communication with the servers, the servers utilizing a striped file system for storing data.

2. An apparatus as described in Claim 1 wherein each server has a network element and a disk element.

3. An apparatus as described in Claim 2 wherein each disk element has a virtual file system with the virtual file system of each disk element together forming a striped VFS.

4. An apparatus as described in Claim 3 wherein all disk elements for a virtual file system act as meta-data servers.

5. An apparatus as described in Claim 4 wherein a file has attributes and each server for each file maintains a caching element that stores a last known version of the file attributes and ranges of modification time and change time values for assignment to write operation results.

6. An apparatus as described in Claim 5 wherein each disk element which is not the meta-data server for a virtual file system is an input output secondary.

7. An apparatus as described in Claim 6 wherein ranges of file modification times or file change times are reserved from the meta-data server by the input output secondary.

8. An apparatus as described in Claim 7 wherein the modification and change times in the ranges obtained from the meta-data server are issued to operations already queued at the input output secondary.

9. An apparatus as described in Claim 8 wherein modification and change times in the ranges obtained from the meta-data server are issued to operations received during a window of time after the ranges are reserved from the meta-data server by the input output secondary.

10. An apparatus as described in Claim 9 wherein operations affecting all stripes of a file begin executions first at the meta-data server for a file, and then execute at all input output secondaries, such that operations at the input output secondaries wait only for already executing operations that have already finished their communication with the meta-data server.

11. An apparatus as described in Claim 10 wherein operations follow one of at least two locking models, the first of which is to synchronize first with the meta-data server, then begin core execution by synchronizing with other operations executing at

the input output secondary, and the second of which is to first synchronize at the meta-data server, and then to synchronize with operations at one or more input output secondaries that have begun core execution at the input output secondaries.

12. An apparatus as described in Claim 11 wherein the cluster network is connected in a star topology.

13. An apparatus as described in Claim 12 wherein the cluster network is a switched Ethernet.

14. A method for data storage comprising the steps of:

creating a file across a plurality of NFS servers;

writing data into the file as strips of the data in the servers, the strips together forming a stripe;

reading strips of the data from the servers; and

deleting the strips from the servers.

15. A method as described in Claim 14 including the step of identifying a disk element for a virtual file system of an NFS server as a meta-data server and disk elements for the NFS severs which are not identified as the meta-data server as input output secondaries.

16. A method as described in Claim 15 including the step of storing in a caching element at each input output secondary for

each active file at a meta-data server a last known version of attributes of the file which are good for a dallying period.

17. A method as described in Claim 16 including the step of storing ranges of modification time and change time values in the caching element for assignment to write operations.

18. A method as described in Claim 17 including the step of making a status request by the caching element to the meta-data server to obtain a file's current attributes.

19. A method as described in Claim 18 wherein the making a status request step includes the step of obtaining modification time and change time ranges from the meta-data server.

20. A method as described in Claim 19 including the step of queuing file read and file write requests at the input output secondary until the file read and file write requests are admitted by the cache element and complete execution.

21. A method as described in Claim 20 including the step of tracking by the cache element of the file read and file write requests executing for the file and the ranges that are being read or written.

22. A method as described in Claim 21 including the step of requesting the cache element move out of invalid node to read mode when a read operation must be executed.

23. A method as described in Claim 22 including the step of checking a byte range affected by a file read request to ensure it does not overlap a byte range of any file write requests previously admitted and currently executing.

24. A method as described in Claim 23 including the step of requesting, in response to a file write request that the cache element move into a write mode.

25. A method as described in Claim 24 including the step of checking with the cache element the byte range affected by the file write request for overlap with any admitted and still executing file read or file write requests.

26. A method as described in Claim 25 including the step, when executing a write request, of allocating a modification time and change time pair from the range of modification times and change times stored in the cache element.

27. A method as described in Claim 26 including the step of checking the head of a queue of pending file read and file write requests to see if a head request can be admitted by the caching element after either a file read or file write request is completed.

28. A method as described in Claim 27 including the steps of detecting by the cache element that a file length must be updated in response to a file write request, moving the cache element into exclusive mode; and making a file write status call to the meta-data server to update length attributes of the file.

29. A method as described in Claim 14 including the step of storing in a caching element at each input output secondary for each active file at a meta-data server a last known version of attributes of the file which are good for a dallying period.

30. A method as described in Claim 14 including the step of storing ranges of modification time and change time values in a caching element for assignment to write operations.

31. A method as described in Claim 14 including the step of making a status request by a caching element to the meta-data server to obtain a file's current attributes.

32. A method as described in Claim 31 wherein the making a status request step includes the step of obtaining modification time and change time ranges from the meta-data server.

33. A method as described in Claim 14 including the step of requesting a cache element move out of invalid node to read mode when a read operation must be executed.

34. A method as described in Claim 14 including the step of requesting, in response to a file write request that a cache element move into a write mode.

35. A method as described in Claim 14 including the steps of detecting by a cache element that a file length must be updated in response to a file write request, moving the cache element into exclusive mode; and making a file write status call to a meta-data server to update length attributes of the file.

36. A method for establishing storage for a file comprising the steps of:

receiving an NFS create request at a network element;

receiving a file create request at a meta-data server from the network element;

allocating an inode number for the file at the meta-data server;

making create calls to input output secondaries to mark the file as allocated by the input output secondaries; and

committing the file create at the meta-data server.

37. A method for removing a file from storage comprising the steps of:

receiving a delete file request at a meta-data server;

removing a file name of the file from a parent directory by the meta-data server at the meta-data server;

putting the file on a file delete list by the meta-data server at the meta-data server;

sending delete calls to the input output secondaries;

receiving at the meta-data server acknowledgment calls from the input output secondaries that they have deleted the file;

deleting the file at the meta-data server;

removing the file from the file delete list; and

placing an inode number associated with the file into a free list by the meta-data server at the meta-data server.

38. A method for reading data in a file comprising the steps of:

receiving an NFS read request for data in the file at a network element;

determining by the network element which VFS stores at least one strip containing the data;

sending a file read request from the network element to at least one disk element of a plurality of servers storing a strip of the data;

obtaining current attributes associated with the file by each disk element;

reading the strips of the file from each disk element having the strips; and

generating a response in regard to the file read request.

39. A method for writing data in a file comprising the steps of:

receiving an NFS write request for a file at a network element;

determining by the network element which VFS is associated with the file;

sending a file write request from the network element to at least one disk element of a plurality of servers having a stripe of the VFS;

acquiring current attributes associated with the file; and

writing a predetermined number of bytes of the data into each VFS strip in succession until all of the data is written into the file.